

TECHNIQUE FOR ENFORCING TEMPORAL UNIQUENESS IN AN
OBJECT/RELATIONAL DATABASE MANAGEMENT SYSTEM ENVIRONMENT

by

Cynthia M. Saracco

5

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates in general to database management systems performed by computers. In particular, it concerns a technique for enforcing temporal uniqueness in an object/relational database management system environment.

10

2. Description of Related Art

Databases are computerized information storage and retrieval systems. An Object Relational Database Management System (ORDBMS) is a database management system (DBMS) which uses relational techniques for storing and retrieving objects. These databases are organized into tables which consist of rows and columns of objects, or data. A database will typically have many tables and each table will typically have multiple rows and columns.

ORDBMS software using a Structured Query Language (SQL) interface is well known in the art. The SQL interface has evolved into a standard language for RDBMS software. SQL has been adopted as such by both the American National Standards Organization (ANSI) and the International Standards Organization (ISO).

All data in ORDBMS software is externally structured into tables. The SQL interface allows users to formulate relational operations on tables. This can be done either interactively, in batch files, or embedded in host language, such as C, COBOL, etc. Operators are provided in SQL that allow the user to manipulate the data. The power of SQL lies in its ability to link
5 information from multiple tables or views together to perform complex sets of procedures with a single statement.

Temporal data is often used to track the period of time at which certain business conditions are valid. Several common forms of data analysis involve evaluating time-related data. Examples are examining customer buying behavior, assessing the effectiveness of
10 marketing campaigns, and determining the impact of organizational changes on sales during a selected time period.

It is often desirable when working with temporal data to ensure integrity for time periods in which conditions will be valid. Unfortunately, conventional techniques used to support uniqueness of non-temporal data fail to work when temporal data is involved. This is because
15 they do not include software needed for verifying and enforcing the temporal uniqueness constraint. For example, a firm may wish to have its DBMS enforce the temporal uniqueness constraint so that "employees cannot earn multiple salaries during the same period of time" or "departments can only report to one division during any given period of time."

In non-temporal DBMS environments, data uniqueness is often guaranteed by defining
20 one or more columns to serve as the primary key and/or defining a unique index on the appropriate columns. However, because commercial object/relational DBMS do not understand

the underlying semantics of temporal data, these techniques cannot enforce temporal uniqueness. Thus, data that is considered invalid from a business point of view can be readily stored in a DBMS environment without error. This would violate the integrity rules that a firm may wish to enforce and render subsequent analysis of its data misleading or useless.

5 While there have been various techniques developed for enforcing temporal uniqueness in a DBMS environment, there is a need in the art for an approach which is cost-effective for vendors to implement and that integrates easily into existing object/relational DBMS environments of customers.

SUMMARY OF THE INVENTION

10 The foregoing and other objects, features, and advantages of the present invention will be apparent from the following detailed description of the preferred embodiments, which makes reference to several figures.

The present invention enforces temporal integrity for valid time data. The technique consists of implementing user-defined functions with templates for triggers and setting
15 constraints to dictate the conditions for the temporal data. Together, these support many forms of temporal analysis and help enforce the fundamental aspects of temporal integrity. Among these aspects are temporal uniqueness, which refers to data that is time period exclusive.

The present invention ensures the integrity of temporal data in object/relational database management systems. Temporal and non-temporal data are organized into data objects in
20 databases, each data object comprised of at least one row and at least one column. A data object

may also be a table containing data. When a user attempts to insert new data into a data object in which temporal uniqueness is a consideration, or when a user attempts to update existing data in a data object in which temporal uniqueness is a consideration, the present invention operates to provide a barrier against inserts or updates that would violate the temporal integrity of the
5 existing data.

All functions include error-checking logic to verify that the start and end points of a time period constitute a valid time period. For example, if the database uses dates to track temporal data and employs a "closed, open" temporal representation, start dates should be less than end dates. Such error-checking logic is built into the functions. However, to avoid populating
10 temporal tables with invalid time periods, it is recommended that CHECK constraints be defined on relevant time-related columns (such as columns that contain date information).

The present invention helps users enforce temporal uniqueness by providing trigger templates and CHECK constraints. One preferred embodiment includes a combination of CHECK constraints and triggers. Since determining the correct logic to implement and
15 successfully code this logic can be a rather challenging and error-prone experience for many users, the present invention simplifies the process by providing templates in a sample database. By substituting appropriate table and column names for those provided with the sample database, users can easily instruct the DBMS to enforce temporal uniqueness for those tables in which they deem such a constraint to be important.

The present invention may also be packaged as a DBMS system extender. The extender may be provided to users of ORDBMS systems desiring to enforce temporal integrity of data within their system environments.

BRIEF DESCRIPTION OF THE DRAWINGS

5 Figure 1 illustrates a block diagram overview of the system for enforcing integrity of temporal data usable in the present invention;

Figure 2 is a table of common operators that may be used in conjunction with one aspect of the present invention;

Figure 3 is a table of user-defined functions that may be used in conjunction with one
10 aspect of the present invention. Figures 3(A) through 3(F) are scalar user-defined functions, while Figures 3(G) and 3(H) are table user-defined functions;

Figure 4, consisting of Figures 4(A), 4(B), 4(C) and 4(D), show update trigger scenarios illustrating conditions by which the present invention operates within the database management system environment; and

15 Figure 5, consisting of Figures 5(A), 5(B) and 5(C), are a continuation of the triggers of Figure 4, showing additional update trigger scenarios which may occur within the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

In the following description of the preferred embodiments reference is made to the accompanying drawings which form the part thereof, and in which are shown by way of illustration specific embodiments in which the invention may be practiced. It is to be understood
5 that other embodiments may be utilized and structural and functional changes may be made without departing from the scope of the present invention.

The present invention discloses a system, method and extender for enforcing temporal data integrity in object/relational database management system environments. The preferred embodiments of the present invention relate in general to database management systems
10 performed by computers, and in particular to a technique for enforcing temporal uniqueness in an object/relational DBMS environment. The present invention provides a trigger-based solution which is a relatively inexpensive development alternative. It affords the user complete control over deployment, and allows users to customize any action taken automatically by the DBMS. It is based on a combination of trigger-based logic, user-defined functions, and a routine to
15 implement CHECK constraints.

This approach provides users with maximum flexibility by enabling them to determine which tables, if any, should be the subject of temporal uniqueness constraints. Users may also customize DBMS actions to be taken if a violation is detected, and inspect SQLSTATE errors to determine the exact nature of the condition that would cause a temporal integrity violation to
20 occur, so that subsequent modifications can be made to avoid the problem in the future.

It should be noted that although many descriptions of this invention contained herein refer to temporal data in terms of dates or points in time, other units of measure, or other levels of temporal granularity, can be supported. These include but are not limited to more precise units of measure, such as timestamps, as well as less precise units of measure, such as data based on
 5 quarterly or annual information. However, since date-level granularity is arguably easier to understand and is the preferred form of temporal granularity for many business applications, it is a preferred embodiment of this invention and is frequently used to explain subsequent details about this invention.

Figure 1 shows a block diagram representation of one preferred embodiment of the
 10 present invention. Referring to Figure 1, a user of the system provides a condition that the time period start dates must be less than the time period end dates. This is shown in Block 102. The user also provides a set of routines to verify that additions or changes to the data object will not violate temporal uniqueness. This is shown in Block 104. Block 102 may be accomplished using a user-defined CHECK constraint providing the initial condition that temporal integrity is to be
 15 preserved. Block 104 may be accomplished by using the trigger templates provided as part of the invention. The trigger templates verify the subsequent conditions necessary to ensure that temporal integrity is preserved. Next, the user enters new data or updates existing data in a database management system as shown in Block 106. The method of entry may be performed by computerized means, for example at a personal computer or over a workstation platform. When
 20 an insert of new data or an update of existing data is attempted, a time period comparison may be initiated between the attempted data entry or update and the existing temporal data as shown in Block 108. The initiation of a comparison may be in the form of triggers within the database system that execute automatically when an attempt is made. Once an initiation is commenced,

user-defined functions operate to perform the actual comparisons of time period start dates and end dates. This is shown in Block 110. After the time period comparisons are performed, the system disallows an attempted insert or update if temporal integrity is to be violated. This is shown in Block 112.

5 The triggers utilize user-defined functions to test for whether a time period shares a time component with another time period. The user-defined functions utilize several operators. Among these operators are those listed in Figure 2. These operators, known collectively as Allen's operators, determine the appropriate logic function to be performed in the time period comparison by the user-defined function. For example, they perform comparisons between the
10 start dates to determine if time periods meet, overlap, or equal each other. These operators, and others described herein, operate with the user-defined functions when triggered by an attempted insert of new data or update of existing data.

One of the user-defined functions, the SHARES function, is used by each of the triggers to determine whether a time period shares a time component with another time period. The
15 SHARES function is described separately in a co-pending patent application by the same inventor herewith filed January 28, 2000 and entitled Technique For Detecting A Shared Temporal Relationship Of Valid Time Data In A Relational Database Management System. The serial number of this co-pending patent application is 09/494,325. Its description is incorporated herein by reference.

20 Other operators which may be used by the user-defined functions include WITHIN, SHARES, MERGES, INTERSECT and UNION. The WITHIN operator returns a true output if

time period X is wholly or partly contained within time period Y. It combines the EQUAL, DURING, STARTS, and FINISHES operators into one operator. The SHARES operator returns a true output if time period X shares a time component with time period Y. Like WITHIN, it combines the EQUAL, DURING, STARTS, and FINISHES operators into one, and further adds

5 X OVERLAPS Y and Y OVERLAPS X. The MERGES operator returns true if time period X MEETS or SHARES a time component with time period Y. The INTERSECT operator returns the maximum (latest) start date and minimum (earliest) end date of two time periods if time period X SHARES a time component with time period Y. The UNION operator returns the minimum (earliest) start date and the maximum (latest) end date of two time periods if these time

10 period MEET or SHARE a time component.

In one preferred embodiment, each trigger is coded to raise a different SQLSTATE error and disallow attempted changes that would violate temporal uniqueness. However, users can modify this behavior of the trigger-based logic, if desired. This may be accomplished by altering the method in which the database management system operates when an attempt is disallowed.

15 For example, the DBMS could be programmed to override the system if a user was not concerned with temporal integrity by disabling the trigger templates. A user may also wish to bypass the entire system of the present invention to disable the behavior of the triggers when entering new data or updating existing data in a data object. Another method involves formulating relational operations either interactively, in batch files, or embedded in a host

20 language. The SQL interface therefore provides operators that allow users to manipulate data and modify the performance of the system.

The trigger-based logic is coded into several routines. One routine supports INSERT row instructions while other routines support UPDATE row instructions. The INSERT trigger routine tests for a match between the primary key of the new row and primary key values of the existing rows. Furthermore, it uses the SHARES function to check if the time period on the new row
 5 shares any time in common with time periods already in existing rows that have the same primary key value. If one or more rows exist in the data object that would cause both these tests to evaluate as true, a SQLSTATE error (e.g., SQLSTATE 70100) is raised.

In a DBMS that supports temporal data, the primary key signifies data values that should be unique in a data object over time. If a user attempts to insert new data into the data object, the
 10 INSERT trigger begins by comparing the new primary key value with existing primary key values. If a match is found, the trigger initiates additional routines for testing the period start and end dates associated with the new primary key value to ensure that there is no time shared with the start and end dates of any existing data in the data object that contains the same primary key value. If this condition were to occur, it would violate temporal uniqueness and the INSERT
 15 trigger would raise an error condition to prevent the INSERT operation from proceeding.

The UPDATE trigger routines test for a unique type of temporal change that might be expressed in the updated row, and could lead to a temporal integrity violation. A preferred embodiment of the present invention may include several UPDATE triggers testing for different conditions. These are listed in Figures 4 and 5. Another preferred embodiment includes a single
 20 UPDATE trigger consolidating all testing conditions. Consolidation adds complexity to developing, debugging, and maintaining such a trigger.

Figures 4 and 5, consisting of Figures 4(A), 4(B), 4(C) and 4(D), as well as 5(A), 5(B), and 5(C), list the several types of temporal updates which might be expressed in the updated row that could result in integrity violations. Each sub-figure illustrates a different type of update that would trigger a comparison of data. Each such BEFORE UPDATE trigger provided in the software of the preferred embodiment of the present invention is described in Figures 4 and 5. For each description, "n" is a variable referring to the new version of the updated row and "o" is a variable referring to the old version of the row being updated. In addition, "s" refers to the column name containing the start date of the period and "e" refers to the column name containing the end of the period.

Another kind of update attempt that involves the primary key can potentially cause temporal uniqueness to be violated. This involves non-temporal data in the primary key column. Additional trigger logic is provided for updates of non-temporal primary key data such that if an updated row shares a time component with the existing rows in the data object that have the same primary key values, SQLSTATE 70008 is raised.

Other user-defined functions may also be combined with triggers to determine whether various temporal conditions are valid, such as whether one time period shares a time component with another time period. Examples of these user-defined functions are listed in the table in Figure 3. Referring to Figure 3(A), the WITHIN function is satisfied if period 1 is DURING, STARTS, FINISHES, or EQUAL period 2. The specific logical implementation of this function is: if date1 >= date3 AND date2 <=date4, return 1; else return 0. In Figure 3(B), the SHARES function is satisfied if period 1 EQUAL, OVERLAP, STARTS, FINISHES, or DURING period 2, or period 2 OVERLAP, STARTS, FINISHES, or DURING period 1. The specific logical

implementation of this function is: if (date1 < date4 AND date3 < date2) or (date1 = date3 AND date2 = date4) return 1; else return 0.

In Figure 3(C), the INTERSECT_START function is satisfied if the periods SHARE a time component. If they do, the function returns the start date of the intersection of these periods (maximum of date1, date3). Otherwise, it returns null. The INTERSECT_END function in figure 3(D) is also satisfied if the periods SHARE a time component. If they do, the function returns the end date of the intersection of these periods (minimum of date2, date4). Otherwise, it returns null.

The UNION_START function as described in Figure 3(E) is satisfied if the periods SHARE a time component or if the periods MEET. The function returns the start date of the union of these periods (minimum of date1, date3) if the condition is satisfied. Otherwise, it returns null. In Figure 3(F), the UNION_END function is satisfied if the periods SHARE a component or if the periods MEET. The function returns the end date of the union of these periods (maximum of date2, date 4) if the condition is satisfied. Otherwise, it returns null.

The INTERSECT function in Figure 3(G) is satisfied if the periods SHARE a time component, and returns the start and end dates of the intersection of these periods (maximum of date1, date3 and minimum of date2, date4). Otherwise, it returns null. In Figure 3(H), the UNION function is satisfied if the periods SHARE a time component or if the periods MEET. In this case the function returns the start and end dates of the union of these periods (minimum of date1, date3 and maximum of date2, date4). Otherwise, it returns null.

The CHECK constraint software routine is useful for both temporal uniqueness as well as general data integrity. It tests whether the starting or beginning point or date of a period is less than the ending point or date of a period. In addition, the columns considered to constitute the temporal primary key must be defined as NOT NULL. If defined otherwise, temporal
5 uniqueness cannot be guaranteed. Primary key data is data used to signify the beginning of a time period. The CHECK constraint and the trigger functions use the primary key for determining where to begin a comparison of time period data starting and ending points.

In another preferred embodiment of the present invention, the system can be packaged into a DBMS extender for users to plug into existing DBMS environments. The extender
10 includes the system and method of enforcing temporal data. Users desiring to enforce temporal data in their respective system environments can therefore utilize the system or method manually or choose to utilize the packaged system extender for accomplishing temporal integrity.

The foregoing description of the preferred embodiments of the invention have been presented for the purposes of illustration and description. It is not intended to be exhaustive or to
15 limit the invention to the precise form disclosed. Many modifications and variations are possible in light of the above teaching. It is intended that the scope of the invention be limited not by this detailed description, but rather by the claims appended hereto.

WHAT IS CLAIMED IS:

1 1. A system for enforcing integrity of temporal data in a database management system
2 environment, the data including a time component, comprising:
3 at least one data object of existing data, the existing data including a time component;
4 a computerized entry system in which new data can be entered into said data object, the
5 new data including a time component;
6 a constraint specifying that a beginning of a time period occurs before an end of a time
7 period;
8 a trigger function initiating a comparison between said new data and said existing data;
9 and
10 user-defined functions determining whether new data shares a time component with
11 existing data in said data object.

1 2. The system for enforcing integrity of temporal data as claimed in claim 1, wherein the
2 data object is a table.

1 3. The system for enforcing integrity of temporal data as claimed in claim 1, wherein the
2 database management system is object/relational.

1 4. The system for enforcing integrity of temporal data as claimed in claim 1, wherein said
2 data object is organized in at least one row and at least one column.

1 5. The system for enforcing integrity of temporal data as claimed in claim 1, wherein the
2 trigger function comprises an insert trigger function.

1 6. The system for enforcing integrity of temporal data as claimed in claim 5, wherein the
2 insert trigger function comprises a comparison between a new primary key value and an existing
3 primary key value.

1 7. The system for enforcing integrity of temporal data as claimed in claim 1, further
2 comprising modifying the behavior of the trigger function to override a violation of temporal
3 data integrity.

1 8. The system for enforcing integrity of temporal data as claimed in claim 1, wherein an
2 insert of new data is disallowed if new data time periods share a time component with existing
3 data time periods.

1 9. A system for enforcing integrity of temporal data in a database management system
2 environment, the data including a time component, comprising:

3 at least one data object of existing data, the existing data including a time component;

4 a computerized entry system for updating existing data in said data object, the updated
5 existing data including a time component;

6 a constraint specifying that a beginning of a time period occurs before an end of a time
7 period;

8 trigger functions initiating a comparison between existing data and updated existing data;

9 and

10 user-defined functions determining whether updated existing data shares a time
11 component with existing data in said data object.

1 10. The system for enforcing integrity of temporal data as claimed in claim 9, wherein the
2 data object is a table.

1 11. The system for enforcing integrity of temporal data as claimed in claim 9, wherein the
2 database management system is object/relational.

1 12. The system for enforcing integrity of temporal data as claimed in claim 9, wherein the
2 trigger functions comprise update trigger functions.

1 13. The system for enforcing integrity of temporal data as claimed in claim 12, wherein
2 the update trigger functions initiate a comparison of time period start points and time period end
3 points.

1 14. The system for enforcing integrity of temporal data as claimed in claim 9, further
2 comprising modifying the behavior of the trigger functions to override a violation of temporal
3 data integrity.

1 15. The system for enforcing integrity of temporal data as claimed in claim 9, wherein an
2 update of existing data is disallowed if updated existing data time periods share a time
3 component with existing data time periods.

1 16. A method for enforcing integrity of temporal data in a management system
2 environment, the data including a time component, comprising:

entering new data or updates of existing data in a data object, the new data and updates of existing data including time components;

specifying a condition that a beginning of a time period occurs before an end of a time period;

initiating a comparison of period start times and period end times when new data is entered or existing data is updated; and

determining whether new data or updated existing data shares a time component with existing data in said data object.

17. The system for enforcing integrity of temporal data as claimed in claim 16, wherein the data object is a table.

18. The system for enforcing integrity of temporal data as claimed in claim 16, wherein the database management system is object/relational.

19. The method for enforcing integrity of temporal data as claimed in claim 16, wherein initiating a comparison of time period start points and time period end points comprises an insert trigger and update triggers.

20. The method for enforcing integrity of temporal data as claimed in claim 19, wherein the insert trigger compares new data and existing data for a match between the new primary key value and the existing primary key value.

1 21. The method for enforcing integrity of temporal data as claimed in claim 19, wherein
2 the update trigger functions initiate a comparison of time period start points and time period end
3 points for new data and updated existing data.

1 22. The method for enforcing integrity of temporal data as claimed in claim 16, further
2 comprising modifying the behavior of the trigger functions to override a violation of temporal
3 data integrity.

1 23. The method for enforcing integrity of temporal data as claimed in claim 16, wherein
2 an insert of new data is disallowed if new data time periods share a time component with existing
3 data time periods.

1 24. The method for enforcing integrity of temporal data as claimed in claim 16, wherein
2 an update of existing data is disallowed if updated existing data time periods share a time
3 component with existing data time periods.

1 25. The method for enforcing integrity of temporal data as claimed in claim 16, wherein
2 determining whether new data or updated existing data shares a time component with existing
3 data in a data object comprises user-defined functions comparing time periods between new data,
4 existing data and updated existing data.

1 26. A computer-based database management system extender, comprising:
2 a database storage device;
3 temporal data organized into at least one data object and stored on said storage device,
4 the data including a time component;

5 an entry system for new data into said data object and for updating existing data in said
6 data object, the new data and updated existing data including time components; and
7 a system for enforcing integrity of temporal data.

1 27. The system for enforcing integrity of temporal data as claimed in claim 26, wherein
2 the data object is a table.

1 28. The computer-based database management system extender as claimed in claim 26,
2 wherein the enforcing system comprises:

3 a constraint specifying that a beginning of a time period occurs before an end of a time
4 period;

5 trigger functions; and

6 user-defined functions determining whether new data or updated existing data shares a
7 time component with existing data in said data object.

1 29. The computer-based database management system extender as claimed in claim 28,
2 wherein the trigger functions comprise an insert trigger function and update trigger functions.

1 30. The computer-based database management system extender as claimed in claim 29,
2 wherein the insert trigger function initiates a comparison between a new primary key value and
3 an existing primary key value.

1 31. The computer-based database management system extender as claimed in claim 28,
2 wherein the update trigger functions initiate a comparison of time period start points and time
3 period end points.

1 32. A method of performing a computer-based database management system extender,
2 comprising:
3 storing temporal data in a database storage device;
4 organizing temporal data into at least one data object and storing on said storage device,
5 the data including a time component;
6 entering new data into said data object and updating existing data in said data object, the
7 new data and updated existing data including time components; and
8 enforcing integrity of temporal data.

1 33. The method claim 32, wherein the data object is a table.

1 34. The method of claim 32, wherein the step of enforcing integrity of temporal data
2 comprises:
3 a constraint specifying that a beginning of a time period occurs before an end of a time
4 period;
5 trigger functions; and
6 user-defined functions determining whether new data or updated existing data shares a
7 time component with existing data in said data object.

1 35. The method of claim 34, wherein the trigger functions comprise an insert trigger
2 function and update trigger functions.

1 36. The method of claim 35, wherein the insert trigger function initiates a comparison
2 between a new primary key value and an existing primary key value.

- 1 37. The method of claim 35, wherein the update trigger functions initiate a comparison of
- 2 time period start points and time period end points.

ABSTRACT

TECHNIQUE FOR ENFORCING TEMPORAL UNIQUENESS IN AN
OBJECT/RELATIONAL DATABASE MANAGEMENT SYSTEM ENVIRONMENT

A system, method and program extender for enforcing the uniqueness of temporal data in
5 an object/relational database management system is provided. There is a combination of user-
defined functions, triggers and specified conditions to determine if attempts to insert new data or
update existing data in a data object in which temporal integrity is a consideration are valid.
Barriers exist to protect against inserts or updates that would violate the temporal integrity of the
existing data. Also included are ways for a user to modify the behavior of the system or method
10 to override triggers for cases in which temporal integrity is not desired.

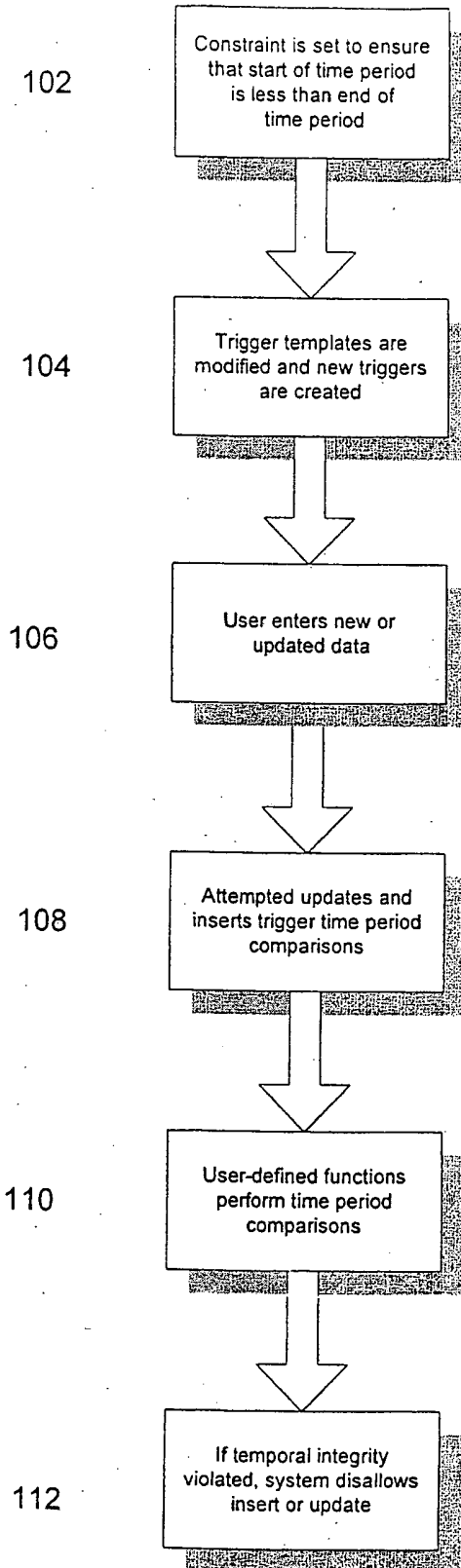


FIG. 1

OPPENHEIMER WOLFF & DONNELLY LLP

Inventor: Cynthia Maro Sacarro

Docket: ST9-99-147

Title: TECHNIQUE FOR ENFORCING TEMPORAL....

Attorney Name: Spyros J. Lazaris, Esq.

Phone No.: (310) 788-5055

Express Mail Label No.: EL502385608US

Sheet 1 of 5

OPERATORS

NOTATION

BEFORE

X BEFORE Y

MEETS

X MEETS Y

OVERLAPS

X OVERLAPS Y

DURING

X DURING Y

STARTS

X STARTS Y

FINISHES

X FINISHES Y

EQUAL

X EQUAL Y

FIG. 2

OPPENHEIMER WOLFF & DONNELLY LLP

Inventor: Cynthia Maro Sacarro

Docket: ST9-99-147

Title: TECHNIQUE FOR ENFORCING TEMPORAL....

Attorney Name: Spyros J. Lazaris, Esq.

Phone No.: (310) 788-5055

Express Mail Label No.: EL502385608US

Sheet 2 of 5

WITHIN(date, date, date, date) returns int

3(A)

SHARES(date, date, date, date) returns int

3(B)

INTERSECT _START(date, date, date, date) returns date

3(C)

INTERSECT _END(date, date, date, date) returns date

3(D)

UNION _START(date, date, date, date) returns date

3(E)

UNION _END(date, date, date, date) returns date

3(F)

INTERSECT(date, date, date, date) returns table (c1 date, c2 date)

3(G)

UNION(date, date, date, date) returns table (c1 date, c2 date)

3(H)

FIG. 3

OPPENHEIMER WOLFF & DONNELLY LLP

Inventor: Cynthia Maro Sacarro

Docket: ST9-99-147

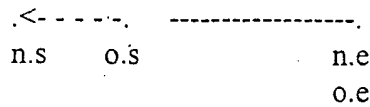
Title: TECHNIQUE FOR ENFORCING TEMPORAL....

Attorney Name: Spyros J. Lazaris, Esq.

Phone No.: (310) 788-5055

Express Mail Label No.: EL502385608US

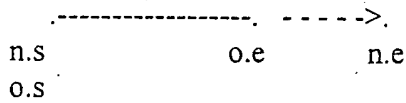
UPDATE WHERE N.S < O.S AND N.E = O.E



SQLSTATE error 70001 raised if updated row shares a time component with existing rows

4(A)

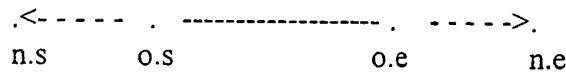
UPDATE WHERE N.S = O.S AND N.E > O.E



SQLSTATE error 70002 raised if updated row shares a time component with existing rows

4(B)

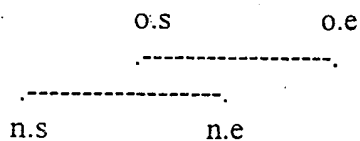
UPDATE WHERE N.S < O.S AND N.E > O.E



SQLSTATE error 70003 raised if updated row shares a time component with existing rows

4(C)

UPDATE WHERE N.S < O.S AND N.E < O.E AND N.E > O.S



SQLSTATE error 70004 raised if updated row shares a time component with existing rows

4(D)

FIG. 4

OPPENHEIMER WOLFF & DONNELLY LLP

Inventor: Cynthia Maro Sacarro

Docket: ST9-99-147

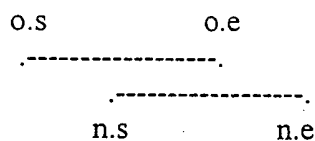
Title: TECHNIQUE FOR ENFORCING TEMPORAL....

Attorney Name: Spyros J. Lazaris, Esq.

Phone No.: (310) 788-5055

Express Mail Label No.: EL502385608US

UPDATE WHERE N.S > O.S AND N.E > O.E AND N.S < O.E



SQLSTATE error 70005 raised if updated row shares a time component with existing rows

5(A)

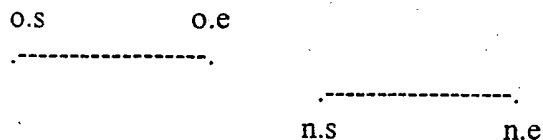
UPDATE WHERE N.S < O.S AND N.E < O.E AND N.E <= O.S



SQLSTATE error 70006 raised if updated row shares a time component with existing rows

5(B)

UPDATE WHERE N.S > O.S AND N.E > O.E AND N.S >= O.E



SQLSTATE error 70007 raised if updated row shares a time component with existing rows

5(C)

FIG. 5

OPPENHEIMER WOLFF & DONNELLY LLP

Inventor: Cynthia Maro Sacarro

Docket: ST9-99-147

Title: TECHNIQUE FOR ENFORCING TEMPORAL....

Attorney Name: Spyros J. Lazaris, Esq.

Phone No.: (310) 788-5055

Express Mail Label No.: EL502385608US

Sheet 5 of 5